KALIDO

Business Modeling

A New Paradigm for Designing and Implementing Enterprise Data Warehouses

October 2005

Bruce Ottmann – Principal Consultant, Kalido

Table of contents

1 Introduction

An enterprise data warehouse provides management with information about what is happening in the business. It must present a single version of the truth from multiple perspectives—and quickly reflect organizational, regulatory, market and other business changes.

A business model is an easy-to-understand representation of things that are of interest to the business (such as business transactions, products, customers, sales people and their inter-relationships). Kalido uses business models to facilitate the creation, operation and management of KALIDO enterprise data warehouses, which provide unique advantages over other data warehousing approaches:

- Data warehouses are **created iteratively**—within 8-12 weeks (versus the 16 months typically required to build an enterprise data warehouse[1]).
- They **rapidly adapt** to changing business conditions or requirements.
- They **preserve historic context** for trend analysis and audit reporting.

These benefits are enabled by a new data warehouse modeling paradigm based on Generic Modeling concepts and principles—an ISO-standard modeling approach that evolved during the 1990s.

This paper describes these Generic Modeling concepts and Kalido's patented approach for leveraging them to rapidly deliver and iteratively evolve flexible enterprise data warehouses containing meaningful and auditable business information. The paper explains what a business model is, how it really works to support change scenarios and contrasts it with other data warehousing design techniques.

Generic Modeling forms the basis for Kalido's business model-driven approach to data warehouse creation and management, enabling data warehouse changes to occur without physical schema changes.

[1] Source: The Data Warehouse Institute

2 The Need for a New Data Warehouse Modeling Paradigm

In a dynamic business, an enterprise data warehouse must:

- Support information about all the business activities we want to measure—in context—independently of the system from which the information comes.
- Draw and reconcile information from autonomous and disparate business units and IT systems despite different codes, formats, definitions, references and data models.
- Rapidly adapt to:
    - Changing business circumstances
    - Source data system changes
    - New information requirements
- Reflect information from the past accurately even if the business structures have changed; for example, comparing information before and after a merger.

Unfortunately data warehouses struggle to meet these objectives:

- The average enterprise data warehouse takes 16 months to implement.
- Enterprise data warehouses cost $2M-$3M USD to develop[2].
- 50% of all custom-built enterprise data warehousing projects are considered failures[3].

A major cause is that the quality of data warehouse modeling methods is poor. It is a basic tenet of good data modeling practice that there should be a clear separation between the different levels of data model—conceptual, logical and physical, but we often find that models at the different levels look remarkably like one another. This gives rise to a number of problems:

- Conceptual model changes normally impact right through to the physical level, usually involving database re-organization, re-programming and delay.
- History is lost. Model changes often cause problems with existing historic data; analysis across the changes at best becomes difficult, at worst impossible. This is true at two levels — both in terms of changes to the schema and in terms of changes to reference data.
- The conceptual model (if it exists) is primarily a view of the data that is to be stored (little different from the physical design) rather than being a model of the business. There is a disconnect between the model, which becomes an IT artifact, and the business users.
- Business rules, specific to how things are done in a particular place, are often fixed in the structure of a data model. This means that small changes in the way business is conducted can lead to large changes in data warehouses.

2.1 The New Paradigm: Generic Modeling

In the 1990s, these shortcomings and their negative affects on information management gave rise to a new *Generic Modeling* paradigm[4] and the development of the Generic Entity Framework, which has evolved into the ISO 15926 standard. The paradigm is based on six fundamental modeling principles:

1. *Entity types should represent, and be named after, the underlying nature of an object, not the role it plays in a particular context.*

[2] Source: Gartner

[3] Source: Gartner

[4] More information can be found at the ISO web site for Part I and Part II of the standard. Visit the Modeling Principles working site maintained by Matthew West and Julian Fowler

2. *Entity types should be part of a subtype/super-type hierarchy in order to define a universal context for the mode.*
3. *Activities and associations should be represented by entity types (not relationships)*
4. *Relationships (in the entity/relationship sense) should only be used to express the involvement of entity types with activities or associations.*
5. *Candidate attributes should be suspected of representing relationships to other entity types.*
6. *Entity types should have a single attribute as their primary unique identifier. This should be artificial, and not changeable by the user.*

Generic Modeling forms the basis for the KALIDO business model-driven approach to data warehouse creation and management. KALIDO adopts these principles in its internal design and this enables data warehouse changes to occur without physical schema changes.

3 Business Model Characteristics—What Makes them Unique

A Business Model is a conceptual model of how we view the real-world things with which the business deals. It is a model of your business that you define, not a pre-defined model that someone else thinks represents your business, and the underlying generic modeling principles from above give rise to its distinctive form. A business model is:

- Able to support the required information
- Clear and unambiguous to all (not just technical people)
- Flexible in the face of changing business practices
- Time-variant (i.e., automatically records its own change history)
- Reusable across the business
- Able to integrate models from different sources
- Able to reconcile conflicting data models

And lastly: *it does not result in database changes when the business model changes.* This concept is what gives KALIDO its unique and patented ability to create and change data warehouses so rapidly. The next sections define all the metadata and reference data components used to create business models, and how they are stored in KALIDO.

3.1 What is a KALIDO Business Model?

The business model for a data warehouse represents the things of interest to the business:

- Things the business "does" (i.e., business activities such as sales, purchases, etc.)
- Measurements we take about these activities
- The context within which we want to view this information (i.e. customers, locations, etc.)

The diagram in Figure 1 shows a KALIDO business model. It can be understood by technical and business people alike. The model hides a great deal of technical complexity but no-one, including the technical people, has to deal with the complexity–KALIDO does that for you.

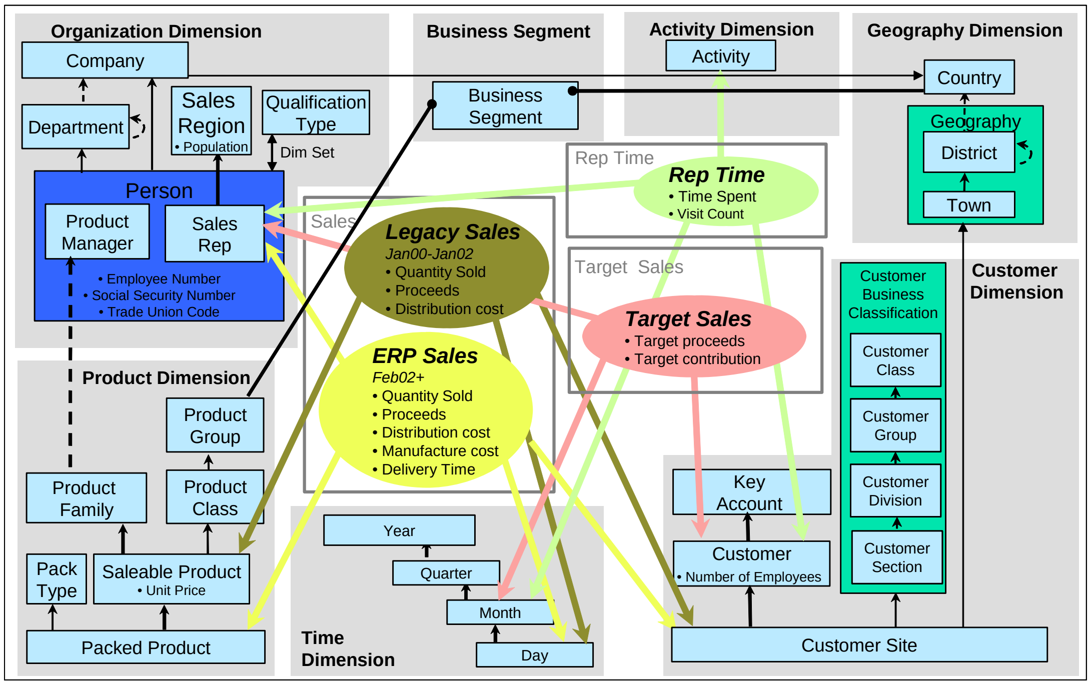


Figure 1: KALIDO Business Model Example

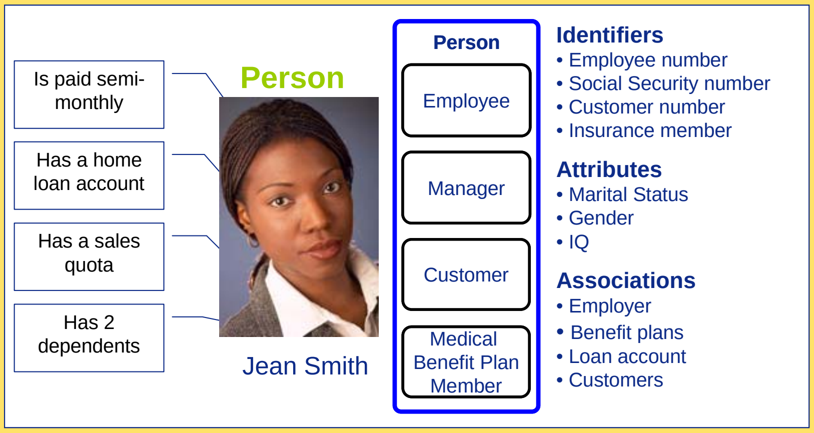
3.2 Elements of a KALIDO Business Model Diagram

- Ovals in Figure 1 represent business transactions—the events that occur in the business (similar to fact tables in traditional data warehouse designs). Text in the ovals identifies the measurements we take about the business transactions. They are similar to columns in a traditional data warehouse fact table but are objects in their own right. Note that "Quantity Sold" is in two transaction datasets. It represents the same business measure.
- Each blue box is a Class of Business Entity (CBE), i.e. a type of real-world thing such as a customer, product, country, etc. This is the same as entity types in conceptual modeling. Bulleted text within CBE boxes identifies attributes (such as a Person's employee number).
- CBEs can define super-types. The super-type Person CBE has two subtypes, meaning some people are Sales Reps, some are Product Managers and others are neither. Subtypes can be nested and a variety of rules about their cardinality can be defined.
- Business Entities (BEs) are representations of real-world objects (i.e., instances of CBEs). They are not shown in the model but it is critical to understand them to understand the model. They are the actual customers, products, countries etc. The inset on the next page describes what is meant by a real-world object.
- Green boxes represent "coding structures." Customer Business Classification is a 4-level unbalanced hierarchy, and a Customer Site can reference any level in the hierarchy. This enables customer classes to have 0,1,2 or 3 more detailed classes below them.
- Gray boxes represent dimensions which group together similar objects.

Real-World Objects in the Business Model

The first Generic Modeling principle states that *"Entity types should represent, and be named after, the underlying nature of an object, not the role it plays in a particular context."* In other words, business model objects should be real-world things that business people understand, like products, customers and locations.

The woman below is a real-world person who has a job as a mortgage loan sales executive, and she may be represented differently in a number of systems (e.g., Personnel, Payroll, Benefits and CRM (she's also a customer))—each with different and perhaps inconsistent attributes. Each system uses different unique codes to identify her. Her Marital Status attribute may not be relevant in the CRM system, but Gender may be held in all systems. Her manager reporting relationship may be mandatory in the personnel system but non-existent in all the others. Providing a system to understand all these representations with any number of attributes and relationships is what the KALIDO business model defines.



Building models in this way enables the business user to understand the model and thus see how information can be provided.

3.3 Business Model Relationships

Now we have described what the boxes and ovals represent, let's look at how the business model defines relationships between them and what the model tells us about the business:

- In Figure 1, thick colored arrows linking the business transactions (ovals) with CBEs (blue boxes) are the dimensions of the transactions, e.g. Each Target Sales transaction is about one Customer, one Month and one Sales Rep.
- Black arrows define the associations between CBEs. They generally represent groupings of objects and so can be thought of as hierarchies. For readability, parents are positioned above children in the model. While this appears to be a fixed hierarchy, these associations are constructed so all the historic relationships or even future planned relationships are remembered. Dashed arrows represent optional associations. Arrows originating from and pointing to the same CBE, represent involutions (e.g., a Department can belong to another Department).
- CBEs can have multiple associations with the same parent, e.g. a Person, such as a contractor, can belong to one company but work for a department in another.
- Unlike most dimensional models, relationships can cross from one dimension (gray box) to another (e.g., the relationship between Product Family and Product Manager). These cross-dimensional relationships enable objects to be shared between multiple dimensions, (e.g. the geography dimension contains groupings for objects in both the Customer and Organization dimensions).
- Most relationships are hierarchical associations as this enables effective roll up of business information, and they are positioned on the diagram with the hierarchy having higher-level objects above the lower-level ones. In addition, many-to-many relationships are supported (e.g., between Person and Qualification Type).

3.4 Business Model Inferences

Here's what the KALIDO business model tells us about our information and how it can support the business:

- Sales data in this model came from one source system before February 2002, and a different source afterwards. The business model gives sufficient information to know that Quantity Sold and Proceeds can be reported by Month, Saleable Product and Customer Site across that period. However if you want more detailed information such as by Day and Sales Rep it is only available since February 2002.
- If you want to report Sales Proceeds versus Target Proceeds, the model shows you that this can be reported by Month, Customer and Sales Rep from February 2002.
- It shows the data needed to load Rep Time (Sales Rep reporting time) into the data warehouse includes Time Spent, Visit Count, one of the Sales Rep ID, Activity ID, Customer ID, and Month.
- If you want to load reference data for a new customer site then you will need an identifier and a name for the site, as well as references to the Customer, Industry Class and Geography. As these are mandatory associations you must provide them. When associations are optional, such as between Product Family and Product Manager you can choose to provide references or not.

3.5 Business Models are Time-Variant

The model is presented as a snapshot making it easy to understand. However, all business model elements are time-variant, i.e. KALIDO automatically assigns effective dates for them as the model changes. We call this the "Corporate Memory" as it ensures a fully auditable history of the whole data warehouse.

- Attributes are stored with effective dates to enable easy reporting of changing values.
- All associations are time-stamped so the history of a grouping's changing contents can be reported (e.g. which customer sites a customer has owned).
- In KALIDO, you can even set the clock back and see how the business model looked at an earlier time and see the business entities that were stored at that time.

3.6 How Business Model changes are Stored – Metadata as Data

In traditional data warehouse models, changes made at the conceptual level often translate to changes to the logical and physical layers as well (causing the high cost and complexity of data warehouse change). In KALIDO, model changes are limited to the conceptual level. Rather than defining a business-specific schema for storing customers and brands, etc., the logical and physical layers are stable, pre-defined schemas for storing the metadata components that define your business model, along with the reference data it describes.

The diagram below illustrates the conceptual, logical and physical layers of the KALIDO Adaptive Data Store—the patented implementation of the generic design principles and mechanism for storing business model changes. Changes to the model are made at the Business Model level as the business context and requirements change. Below that the schema is highly stable—it is designed to store the business model as highly abstracted entities.

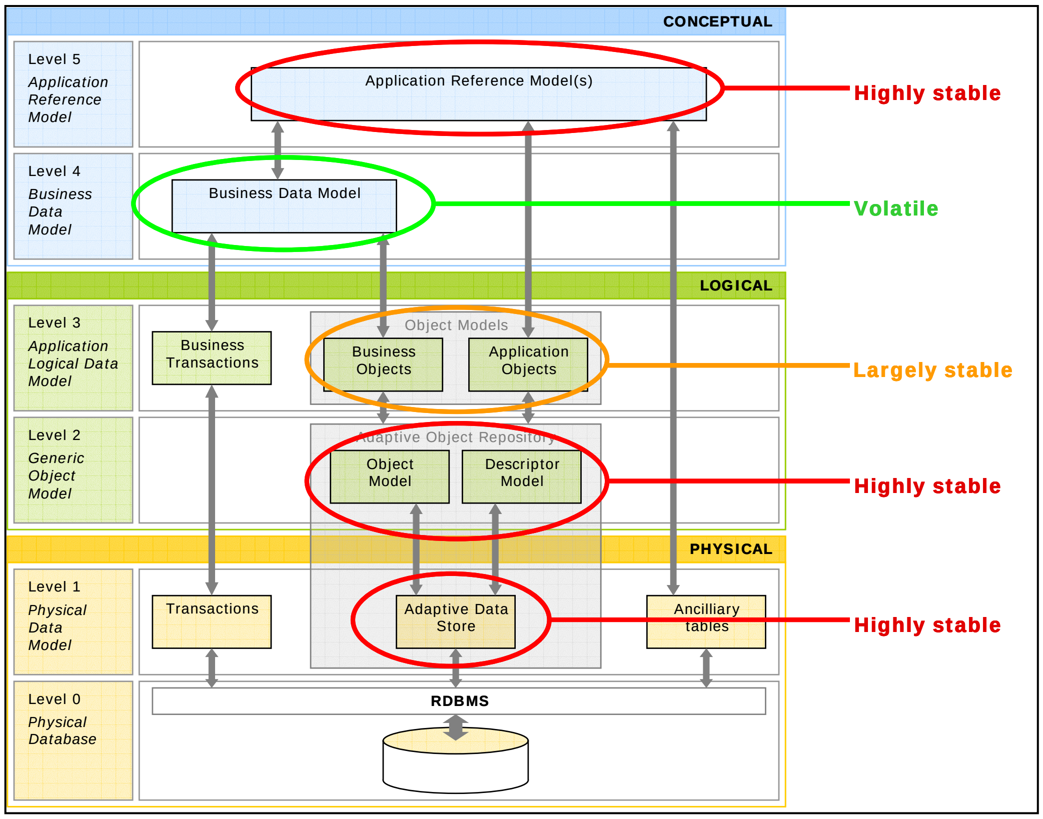


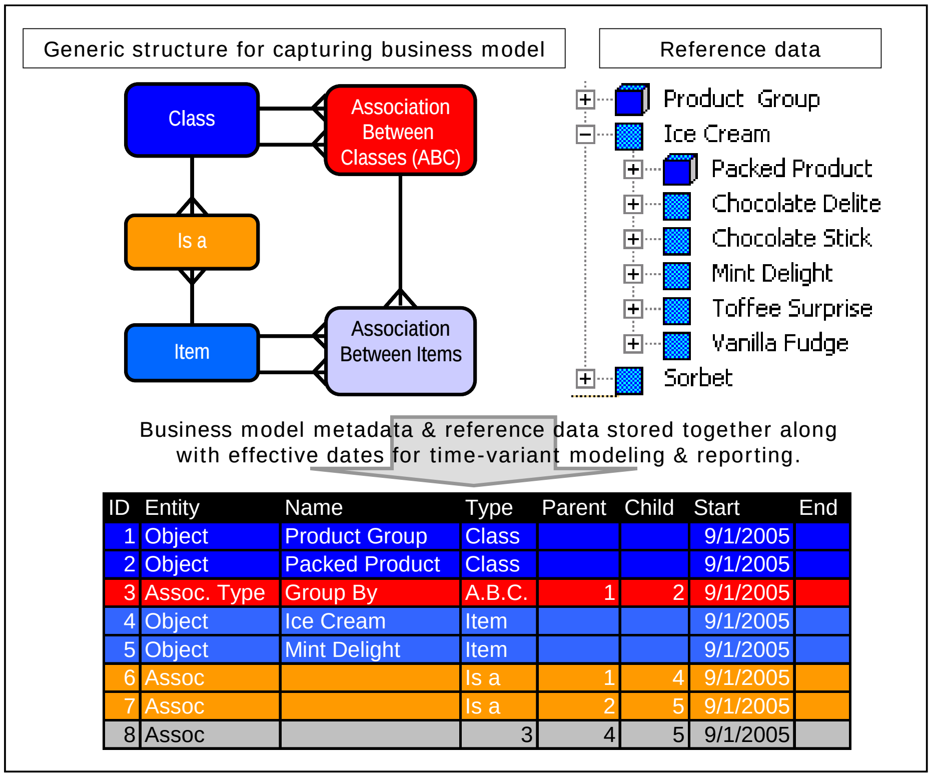
Figure 2: Conceptual, logical and physical layers of the KALIDO Adaptive Data Store

The generic modeling approach provides the ultimate level of model abstraction by identifying and storing every piece of information as an object in its own right (Levels 0 and 1). This enables any number of attributes and relationships to be defined for each individual entity. The physical design for your data warehouse is already defined even though you may not yet have implemented KALIDO.

At Level 2 there are two general-purpose models, the Object model and the Descriptor model. Metadata objects such as CBEs, Measures and Transaction Datasets as well as individual Business Entities (BEs) all conform to the Object Model. All attributes, names and identifiers conform to the Descriptor Model. It is easy to add, remove or change these things by linking in a new object or descriptor into the system. A meta object property is represented in the same exact way as an attribute of a BE.

At Level 3 the objects are more specific and separate models are defined for each type of object supported by the data warehouse – so there is a separate model for a CBE, another for a Measure and yet another for a BE. There are also some additional models at this level for more general items such as users and security access control lists.

The business model is defined in terms of these Level 3 objects and is mapped to the physical design through all these levels. Finally, there is the Application Reference Model which defines the KALIDO data warehousing and master data management applications making them perform the functions they are designed for (e.g., data that drives behavior of the KALIDO applications, wizards, etc.)



Business model metadata & reference data stored together along with effective dates for time-variant modeling & reporting.

ID	Entity	Name	Type	Parent	Child	Start	End
1	Object	Product Group	Class			9/1/2005	
2	Object	Packed Product	Class			9/1/2005	
3	Assoc. Type	Group By	A.B.C.	1	2	9/1/2005	
4	Object	Ice Cream	Item			9/1/2005	
5	Object	Mint Delight	Item			9/1/2005	
6	Assoc		Is a	1	4	9/1/2005	
7	Assoc		Is a	2	5	9/1/2005	
8	Assoc		3	4	5	9/1/2005	

Figure 3: KALIDO business model storage

The diagram above shows that business model metadata is held in KALIDO as a set of data records–the same way that it handles business reference data (e.g. products and customers). Structurally, both obey the same rules and this provides some special abstraction capabilities. Data warehouse behavior in KALIDO is altered by changing data in the Adaptive Data Store table diagramed above. This radically impacts data warehouse development and management:

- Generic modeling enables a high degree of flexibility making change possible without having to re-develop the physical model or ETL programs.
- Changes are implemented by end-dating one record and creating a new record to reflect the change, thus maintaining all history whether it is an attribute value or an association between two entities.
- As the business model metadata and reference data are related, any business model change that would invalidate the reference data can be prevented (see inset) just as entering any data that does not conform to the business model can be prevented.
- It fosters an iterative data warehouse development approach enabling it to evolve as the business evolves. This is important because:
  - People often don't know what information they need at project start.
  - Pressure to capture requirements completely results in lengthy analysis.
  - Business people change their minds about how to view information based on changing events in the business.
  - There are always new requirements to include in the data warehouse.

The Kalido Approach is to "think big, start small, iterate and evolve." KALIDO business modeling makes this really possible rather than a great but impractical theory.

3.7 Reporting on KALIDO Data Warehouses

If KALIDO stores its business model metadata and reference (i.e., dimension) data together in a single table, you may wonder how information is queried and how good is performance? The answer to this lies in the star and snowflake schema designs. KALIDO automatically generates a 'data warehouse reporting schema'—de-normalized tables similar to dimension tables (called Mapping and Attribute tables) with all the time-variant and multi-granularity capability included. Query generation is defined by the business model but runs from the Transaction, Mapping and Attribute tables. Model changes may change the format of these tables but as they are automatically generated from the business model there is no programming required.

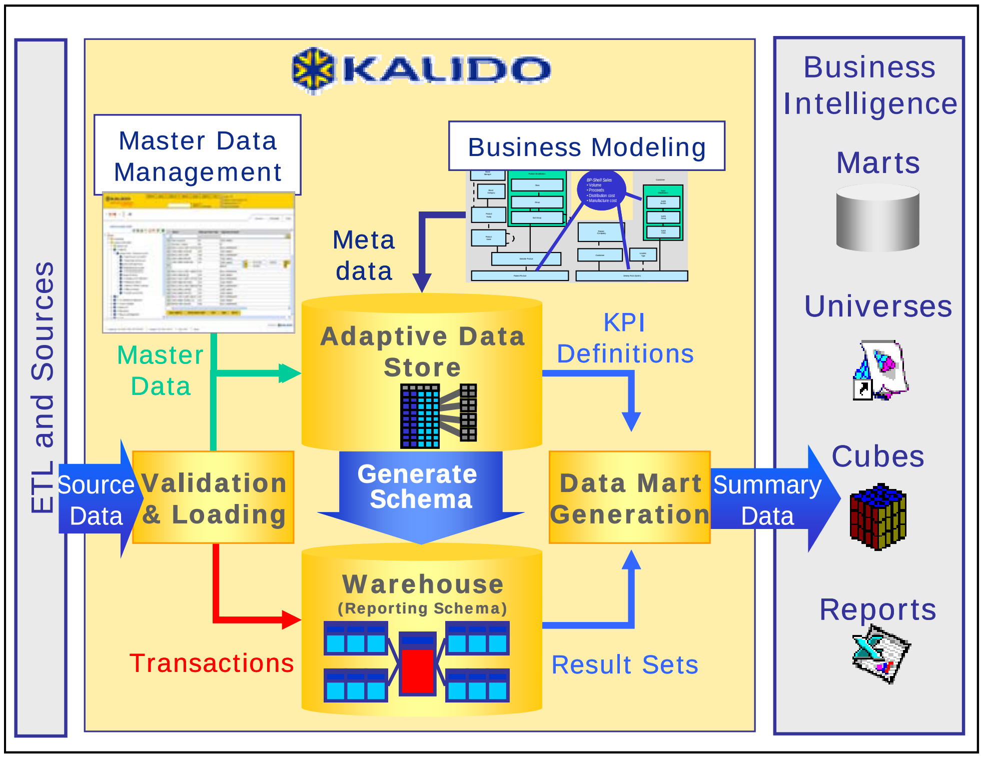
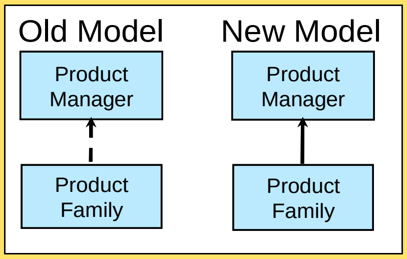


Figure 4: Business Models drive the generation and management of enterprise data warehouses

Data Warehouse Information You Can Trust

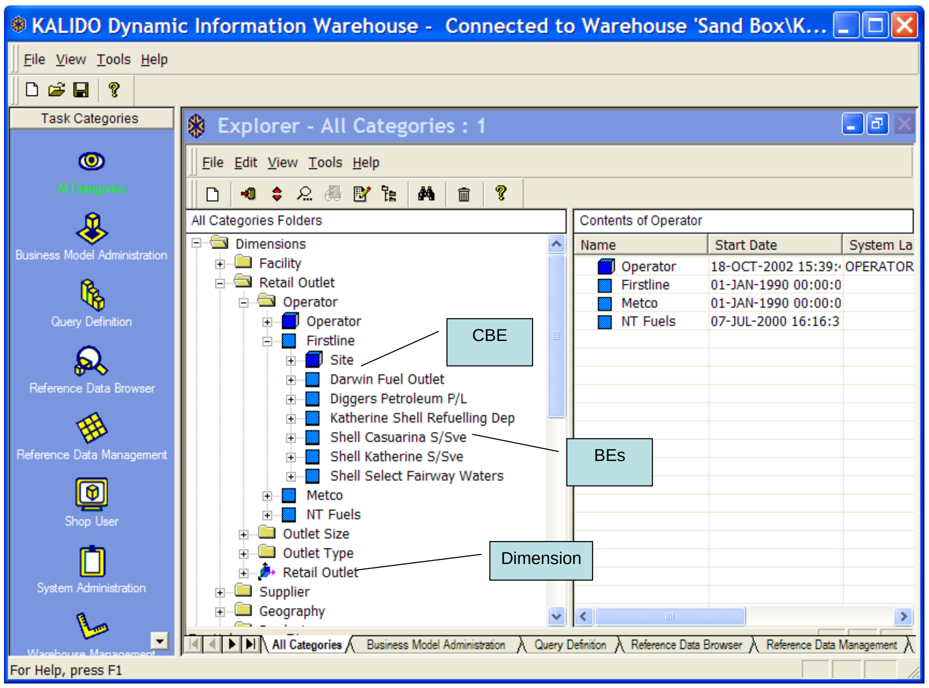
The reference and transaction data loading functions in KALIDO never load data that does not conform to the business model. But what if a model change would invalidate information already in the data warehouse?



This simple model change is dependent on the data stored. A relationship between a Product Family and a Product Manger is defined as optional and some Product Families have a Product Manager and some don't.

If you make this a mandatory relationship, you first must ensure every Product Family has a Product Manager and only then will KALIDO allow the model change. Once it is made, KALIDO enforces the model rules on new data. You can always trust the information in a KALIDO data warehouse to conform to the business model.

4 Implementing and Using Business Models in KALIDO® 8



```
Dimensions
  Facility
  Retail Outlet
    Operator
      Operator
      Firstline
        Site
        Darwin Fuel Outlet
        Diggers Petroleum P/L
        Katherine Shell Refuelling Dep
        Shell Casuarina S/Sve
        Shell Katherine S/Sve
        Shell Select Fairway Waters
      Metco
      NT Fuels
    Outlet Size
    Outlet Type
    Retail Outlet
  Supplier
  Geography
```

Name	Start Date	System La
Operator	18-OCT-2002 15:39::	OPERATOR
Firstline	01-JAN-1990 00:00:0	
Metco	01-JAN-1990 00:00:0	
NT Fuels	07-JUL-2000 16:16:3	

Figure 5: The business model as it's seen and managed in KALIDO DIW

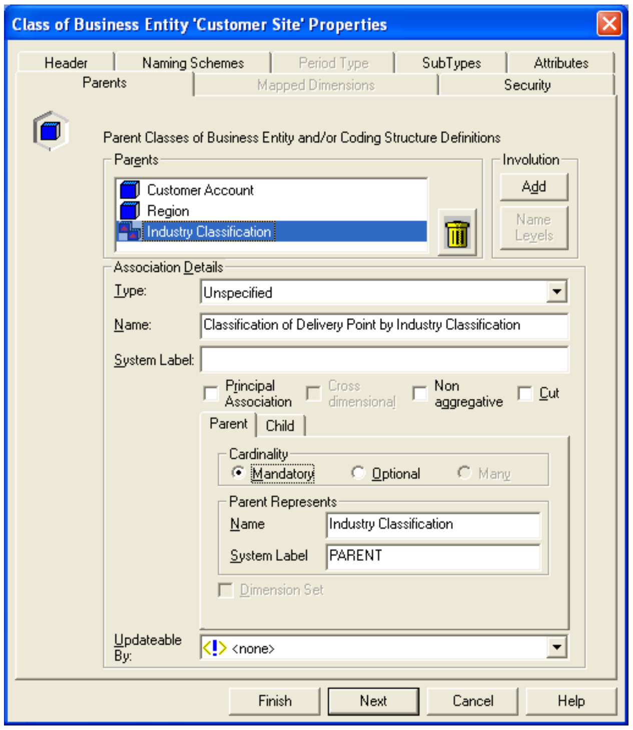
The KALIDO 8 enterprise data warehousing and master data management software suite includes a GUI wizard-driven application (KALIDO DIW) for defining and using business models to drive the creation and management of data warehouses. Business models can also be imported or exported using a special XML format. Figure 5 shows the business modeling interface in KALIDO DIW and highlights how some of the different parts of the business model are represented. From this screen all the functions necessary to develop and maintain a data warehouse are initiated. This is done using wizards taking you through the requirements, step by step, so that nothing is forgotten.

The following sections show how KALIDO enables you to modify and use business models to drive your enterprise data warehouses. The sections illustrate how:

- Classes of Business Entity are defined
- Business Entities are defined
- Transaction data sets are defined
- Data load files are mapped to the business model for import
- The business model drives the definition of queries (for generating data marts, Excel pivot tables, BusinessObjects Universes, etc. from the KALIDO data warehouse)

<u>Defining Classes of Business Entity (CBE)</u>

A CBE always has 2 attributes – an ID code and name, plus optional characteristics such as: additional identifiers, numeric and text attributes, relationships with other CBEs, etc.



Parents: Customer Account; Region; Industry Classification

Type: Unspecified

Name: Classification of Delivery Point by Industry Classification

System Label:

☐ Principal Association ☐ Cross dimensional ☐ Non aggregative ☐ Cut

Cardinality: [✓ Mandatory] Optional Many

Parent Represents Name: Industry Classification

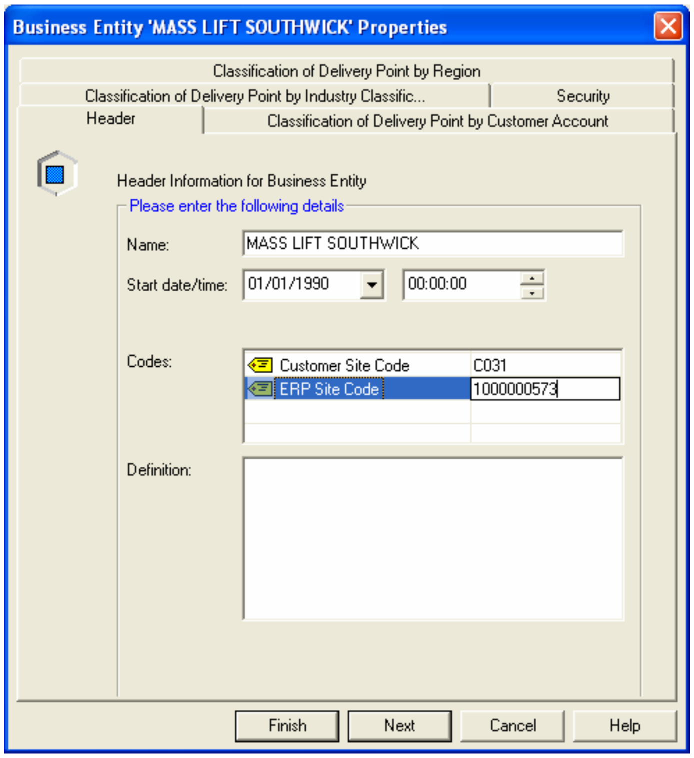
System Label: PARENT

☐ Dimension Set

Updateable By: <none>

Figure 6: The KALIDO CBE Editor Wizard

The panel in Figure 6 describes the associations between CBEs and is only required for CBEs with parents. For the simplest CBE only the Header panel is required meaning a CBE can be set up and used within seconds



Business Entity 'MASS LIFT SOUTHWICK' Properties

Classification of Delivery Point by Region
Classification of Delivery Point by Industry Classific...
Security
Header
Classification of Delivery Point by Customer Account

Header Information for Business Entity

Please enter the following details

Name: MASS LIFT SOUTHWICK

Start date/time: 01/01/1990 00:00:00

Codes:

Customer Site Code	C031
ERP Site Code	1000000573

Definition:

Figure 7: The KALIDO BE Editor Wizard

<u>Defining Business Entities (BE)</u>
Once a CBE is defined then the BE Wizard, as shown in Figure 7, can be used to enter reference data.

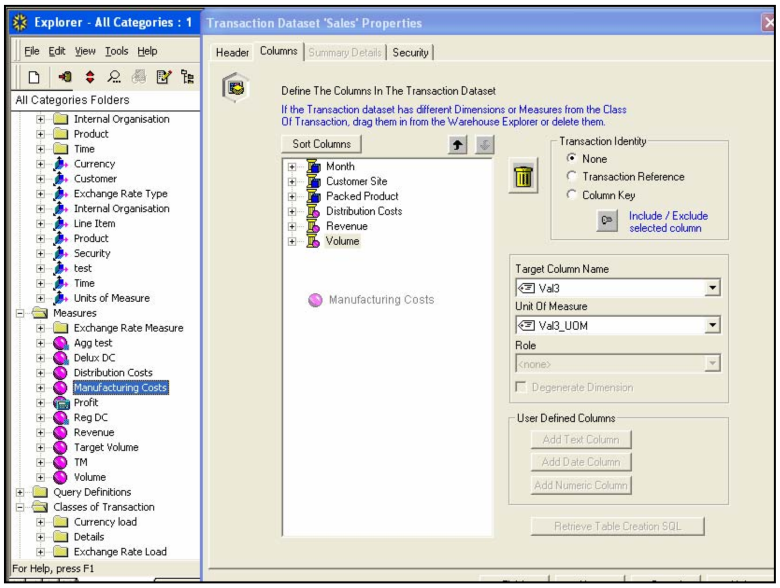
The wizard is defined by the CBE definitions with tabs for each of the parent CBE associations.

Where multiple identifiers are defined these are displayed on the Header panel.

Typically, programs are automatically generated by KALIDO to batch-load higher volumes of reference data from other systems.

<u>Defining Transaction Datasets</u>
A transaction dataset defines how transaction data will be held in KALIDO and is defined once the CBEs and Measures are created.

It requires just 2 panels, including the one shown in Figure 8, to be completed by dragging CBE and Measure objects into the wizard. Afterwards, a File Definition is created, which defines how data held in an external file maps to the transaction dataset for loading purposes.



Explorer - All Categories : 1

All Categories Folders

- Internal Organisation
- Product
- Time
- Currency
- Customer
- Exchange Rate Type
- Internal Organisation
- Line Item
- Product
- Security
- test
- Time
- Units of Measure
- Measures
  - Exchange Rate Measure
  - Agg test
  - Delux DC
  - Distribution Costs
  - Manufacturing Costs
  - Profit
  - Reg DC
  - Revenue
  - Target Volume
  - TM
  - Volume
- Query Definitions
- Classes of Transaction
  - Currency load
  - Details
  - Exchange Rate Load

For Help, press F1

Transaction Dataset 'Sales' Properties

Define The Columns In The Transaction Dataset

If the Transaction dataset has different Dimensions or Measures from the Class Of Transaction, drag them in from the Warehouse Explorer or delete them.

- Month
- Customer Site
- Packed Product
- Distribution Costs
- Revenue
- Volume

Manufacturing Costs

Transaction Identity: [✓ None] Transaction Reference Column Key

Target Column Name: Val3

Unit Of Measure: Val3_UOM

Role: <none>

Degenerate Dimension

User Defined Columns

Figure 8: The KALIDO Transaction Dataset Wizard

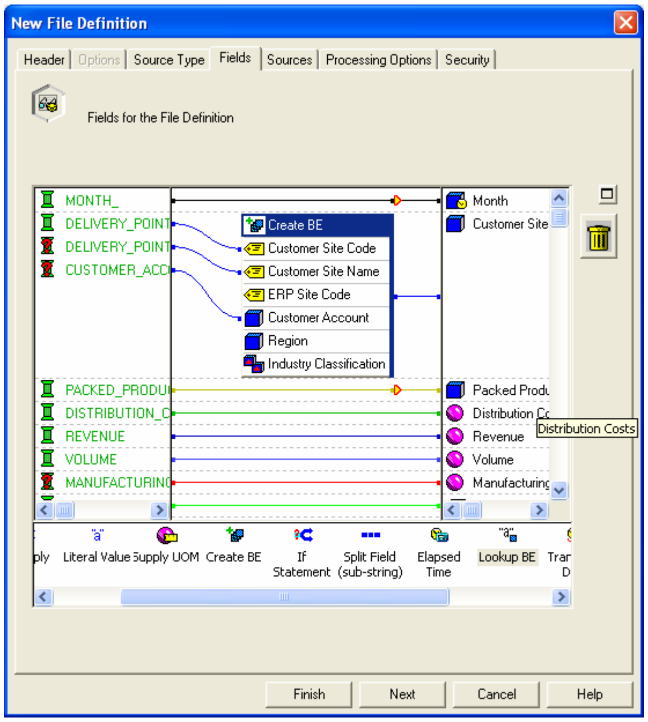


Figure 9: *The KALIDO File Definition Wizard*

<u>Creating File Definitions</u>

The File Definition Wizard (Fig. 9) maps source data to the objects defined in the business model enabling reference data or transaction data to be loaded (and optionally for new BEs to be created on the fly for any new reference data found in the transactions). Mapping rules can be complex with many functions available for manipulating the data as it is loaded into the data warehouse.

These KALIDO mapping facilities do not replace ETL software—they complement it. Refer to the KALIDO white paper "*Best-practice ETL for KALIDO 8*" to learn how to best divide work between KALIDO and ETL software solutions.

<u>Defining Queries</u>

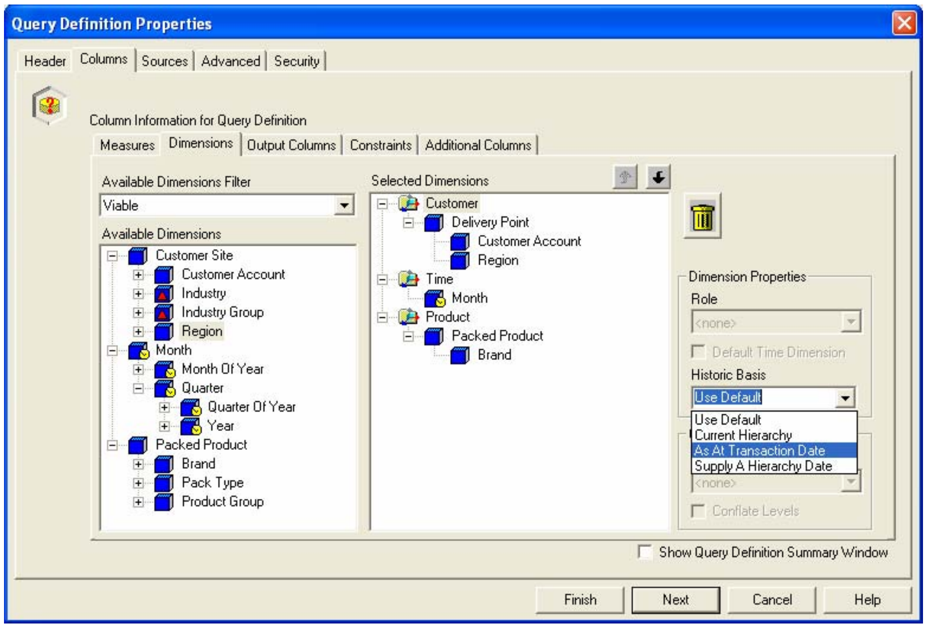


Figure 10: *The KALIDO Query Definition Wizard*

The Query Definition Wizard (Fig. 10) uses the business model to determine which reference data support the delivery of selected measures. It dynamically adjusts as different measures are included in the query. It can also deliver calculated measures, currency conversion, year to date and other relative time periods.

So within a matter of hours a simple data warehouse can be designed and implemented, and because of the rigor involved in the modeling structures and the iterative development approach it enables, this will evolve into an industry-strength data warehouse within weeks.

5 Comparing Data Warehouse Modeling Methods

Using a working example, let us compare how KALIDO business modeling and two alternative design approaches—normalized and star schema designs—suit the needs of an enterprise data warehouse as business requirements evolve. This chapter describes:

- An overview of normalized and star-schema design approaches
- Implementing a simple data warehouse design using the three approaches
- How the three approaches adapt to new business requirements
- How to model attributes
- Making the models time-variant
- Handling unbalanced hierarchies
- Dealing with alternate identifiers of information from disparate systems
- Implementing support for matrixed business segmentations

The following table summarizes the differences between the three data warehouse design approaches and the implications those differences have on data warehouse delivery and maintenance.

Requirement	Normalized Design	Star Schema Design	KALIDO Generic Design
Maintaining and reporting history (of hierarchies and attributes)	Hard - Have to design in	Slowly changing dimension-choose at design time	Automatic
Flexibility	Re-code after change	Re-code after change	No re-coding
Managing master data	Programmed	Very difficult because de-normalized	Data-driven
Cross-dimensional relationships	Not Applicable	Hard	Easy
Variable-depth hierarchies	Hard	Hard	Easy
Subtypes	Hard	Hard	Easy
Querying in SQL	Hard	Limited	Automatically generated
Changing sources	Hard	Hard	Easy
Reporting across changed sources	Hard	Hard	Easy
Handling multi-granularity	Hard	Hard	Easy
Logical design	Major task	Major task	No work
Physical design	Major task	Major task	No work
Complex relationships without double counting	Hard	Medium	Automatic
Making inferences about the most appropriate way to handle data in the data warehouse	Only possible with additional design functionality	Only possible with additional design functionality	Built-in
Handling multiple identifiers	Medium	Hard	Easy

The following sections explain the differences in detail.

5.1 Overview of Normalized and Star Schema Designs

Normalized and star-schema design methods both start with entity-relationship modeling to create conceptual models of the business (similar to KALIDO business models). The models are then used to create systems by first creating logical models of how to represent data about these real-world things. Then physical models are defined for a relational database management system (RDBMS) implementation. Often the conceptual model looks similar to the physical model but with a number of features to enable implementation (e.g. foreign keys, fields for attributes and data types to represent numbers and dates, etc.).

Normalized Design

The normalized approach recommends that data should be analyzed and defined in third-normal form and that this should form the basis of the data warehouse.

Advantages:

- Truly reflects the business with each object defined only once therefore minimizing the impact of any change
- Such models have built-in referential integrity
- Can be understood by business people via careful walkthroughs

Disadvantages:

- Models become very complex even when they cover a relatively small scope
- Often require complex transformations and multiple load programs
- Maintaining historical business context requires complex design and development
- Has poor performance, due to the high number of joins required for queries

Star Schema Design

Business transactions (fact data) are stored in a table surrounded by a set of dimension tables holding reference data. Simple joins enable the fact data to be presented in the context of data in the dimension tables. "Conformed" dimension tables are shared between fact tables, minimizing dimension table maintenance.

Advantages:

- Efficient for fast and easy data retrieval
- Simpler database structures
- Easy to optimize

Disadvantages:

- Reference data is heavily duplicated and when it changes it is critical to change it everywhere or suffer inconsistent results
- Hard to maintain complex relationships
- Lack of referential integrity
- Difficult to maintain historical business context
- Requires complex data validation and loading programs which must be re-worked when the model changes

5.2 *An Initial Sales Data Warehouse Model*

The conceptual model below (Fig. 11) represents a simple sales data warehouse with sales transactions which are for Customer Sites' purchases of Saleable Products on a particular day. We will treat the requirement as the start of a data warehouse which we plan to evolve.

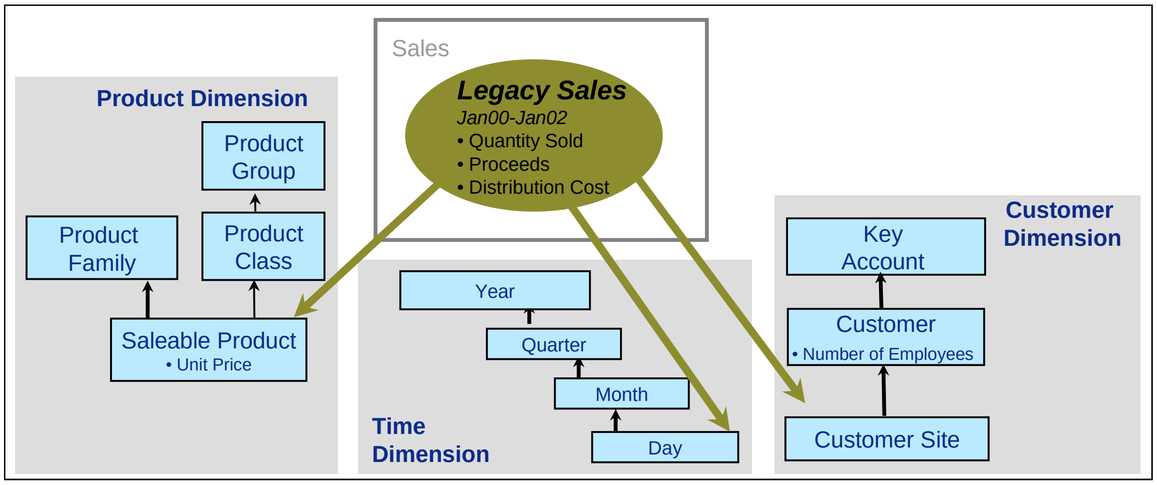


Figure 11: An initial business model

The model shows that business people want to summarize the Quantity Sold, Proceeds and Distribution Costs by Customer, Key Account, Product Family, Product Class and Product Group as well as by Months, Quarters and Years.

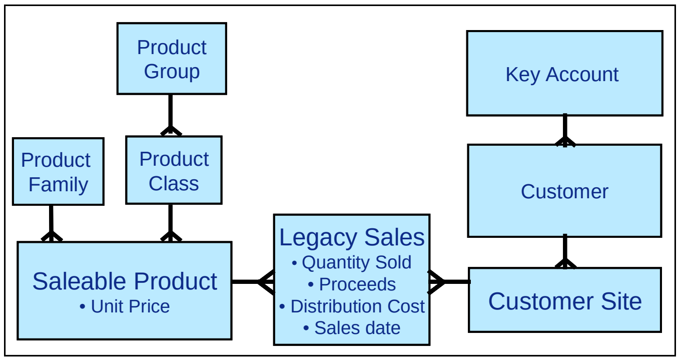


Figure 12: Normalized design

The normalized design (Fig. 12) looks similar to the conceptual model above, except relationships are replaced by foreign key references. It is easy to navigate and report from and has no duplication of master data so it is easy to manage. There are quite a few joins and as the model grows this may hurt performance. Data can be summarized by Month and Year easily by the RDBMS. Quarterly figures may involve more work.

In the star schema (Fig.13), we de-normalize some of the tables making it more efficient to query as there are fewer joins. However, data such as Product Family, Product Class and Product Group have to be duplicated in every Saleable Product record. Similarly Customer and Key Account are duplicated for each Customer Site.

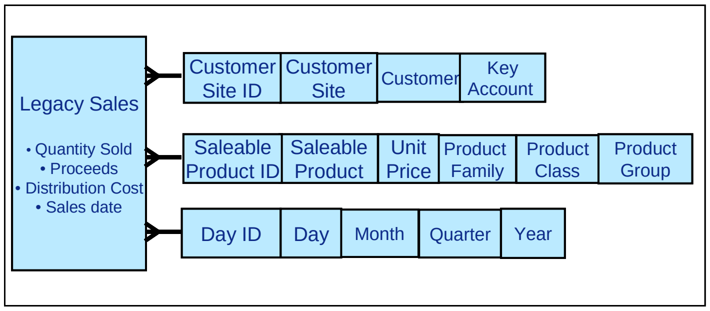


Figure 13: Star-schema design

This presents an issue for the management of the master data (e.g., if the Key Account changes, it has to be changed in all the Customer Site records).

In KALIDO, the business model is the conceptual model and when this model is entered into KALIDO, load programs are automatically available for transaction and reference data loading, mapping and attribute tables are created and query SQL which runs from the mapping and attribute tables will be automatically generated based on the business model.

5.3 Introducing New Business Transactions Types

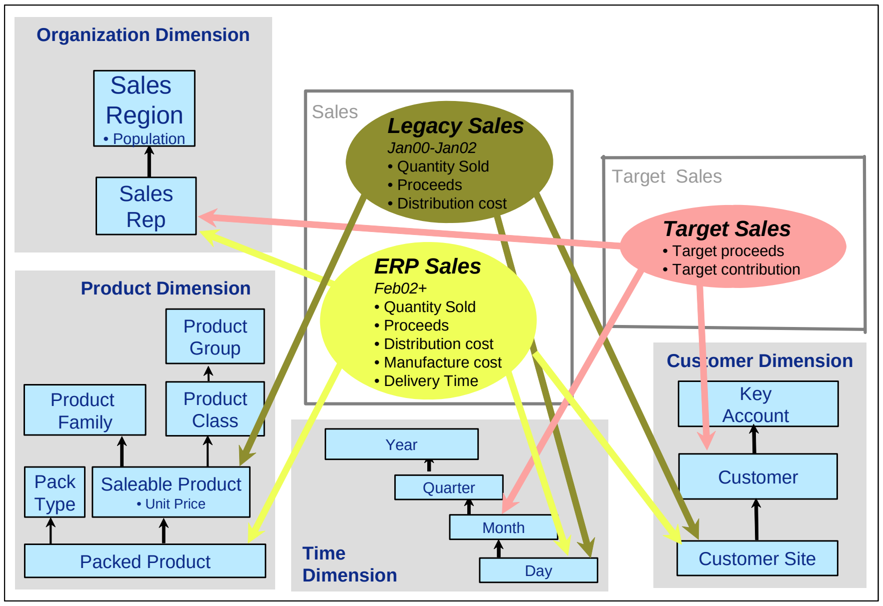


Figure 14: Business model extended with additional transaction types

Suppose we replace the Legacy sales system with a new ERP system and introduce target setting. The ERP system defines Packed Products rather than just Saleable Products allowing us to market multiple product packages. We also add the Organization dimension and target sales data so we can compare sales reps' actual sales with their targets. The business model in Figure 14 reflects these changes.

The normalized design (Fig. 15) follows the conceptual model very closely and enables relative ease of change. However, loading programs need to be re-written with careful validation processes, and the queries will all require additional joins and greater SQL complexity.

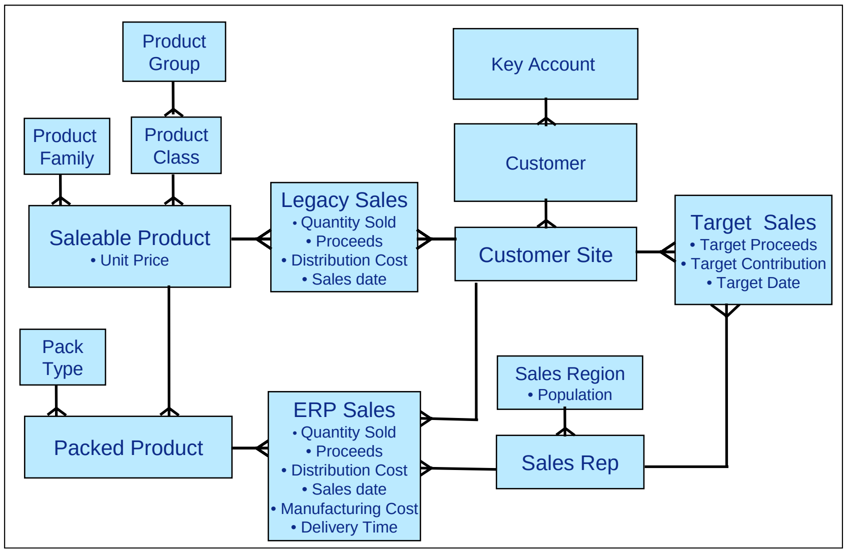


Figure 15: Normalized design extended to meet the new requirements

The star-schema design (Fig. 16) is now considerably more complex. We need conformed dimensions where the dimension tables are shared between the different fact tables. As these have multiple granularities, it is necessary to form surrogate keys that enable the tables to reference either the Customer or Customer Site with Customer attributes.

Keeping tables up to date and consistent is a complex task. Update programs must carry out lots of coded validation. For example, if a company is sold from one Key Account to another:

- All the Customer Site records have to change the Key Account, and
- The Customer record has to change its Key Account

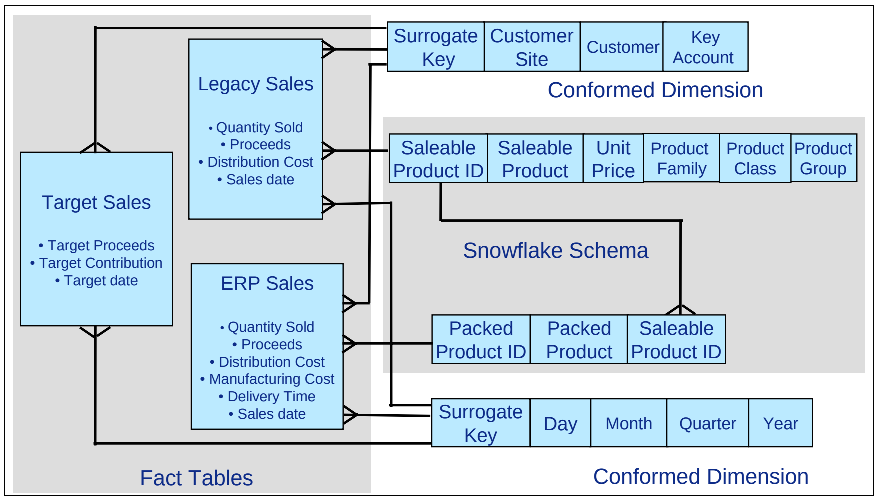


Figure 16: Star-schema design extended to meet new requirements

To add Packed Products, we could use conformed dimensions again or turn the star schema into a snowflake schema. In our example, a new table for Packed Product is created with a reference to the Saleable Product table. This saves changing the Saleable Product load and validation programs but makes the processing more like that of a normalized design. Although still quite efficient, querying this design is now more complex. It can compare Target and Actual Sales and provide seamless information across the Legacy-to-ERP system change.

When new transaction datasets are added in KALIDO, the existing queries which ran on the Legacy Sales transaction dataset will now run on both the Legacy and ERP transaction dataset without any coding required. KALIDO achieves this by changing the mapping and attribute tables in a similar way to the creation of conformed dimensions. The difference is that the KALIDO software does it automatically and manages all the duplication thus giving easy maintenance and good query performance.

5.4 Handling Attributes

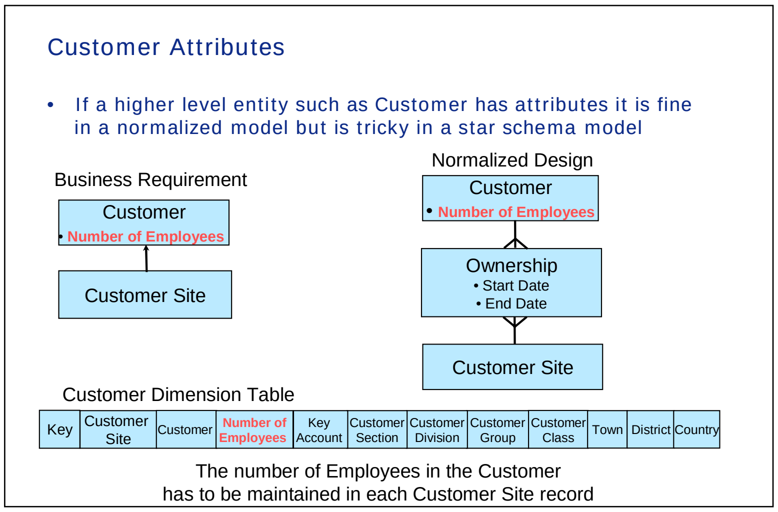


Figure 17: How the 3 approaches handle attributes

Figure 17 shows that adding attributes to higher-level objects such as Customer in a normalized design is done by defining them in the appropriate table. In a star schema, the attribute values must be repeated in every record that refers to the Customer. This has to be managed by the load programs to ensure it is properly maintained. In essence the attribute is deemed to be an attribute of the Customer Site.

In KALIDO, the attribute is added to the customer object and this is the only place it has to be managed much like the normalized design. KALIDO generates mapping and attribute tables to enable efficient reporting of the attribute. These tables look very similar to the dimension tables in a star schema design but with the de-normalization being an automated process.

5.5 Time Variance

Time variance is the ability to remember historic perspectives. The requirement is to be able to know how something was classified or who owned something and how this changes as time passes. The sidebar defines time variance and why it is important to business reporting. Unfortunately it is difficult for data warehouses to do this, and because business people cannot always understand or appreciate the need for it, it is often ignored.

In Figure 18 below, the Customer dimension has been enhanced with new classifications. Also, all Customer relationship and classification changes must be time-variant to enable accurate historic reporting (e.g., Customer Site ownership changes, Customer classification changes, etc. are reflected).

Time Variance—Easy but Painful to Ignore

Future change, be it organizational, regulatory or geographical is hard to conceive. In 1988, who imagined that within a few years, Yugoslavia and East Germany would cease to exist? Enabling a data warehouse to report pre- and post-change information together in a meaningful context is very expensive and time-consuming. Couple that with the pressure to rapidly meet other business requirements, and with the inability for any of us to predict change (especially at system design time!), and you can see why the time-variant reporting requirement is often ignored. But at what expense? The real-life case below illustrates the value of time-variant reporting:

> *A beverage company paid rebates to customers at year-end, based on ownership of customer sites at year end. The sales data warehouse did not reflect customers selling sites to one another throughout the year, resulting in mis-payments and a multi-million dollar customer service dilemma.*

For regulatory compliance and other reasons, data warehouses must remember how things were in the past because at some point business people will expect to be able to be report on them that way. Therefore, KALIDO automatically enables you to perform time-variant reporting on your business model as it changes.

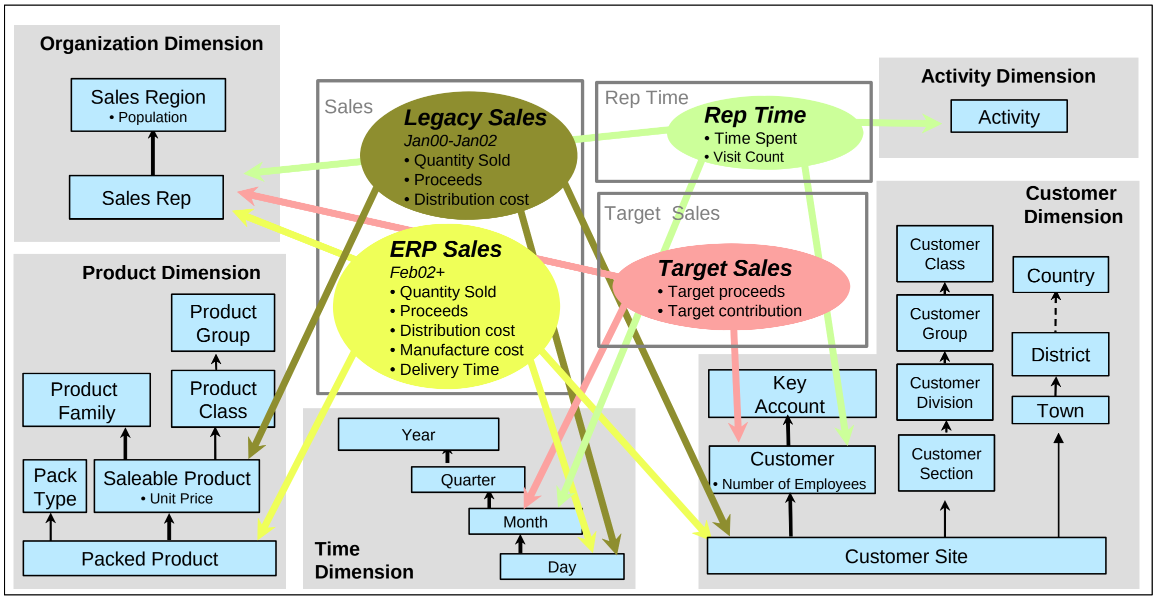


Figure 18: The business wants to track Customer Classifications and how they change over time

Time-variant Customer information is represented in a Normalized model by defining associations with start and end validity dates between all things related to Customer. To find out which Customer owned a specific site on a specific date, for example, a query has to check the validity dates. This complicates the diagram (Fig. 19), making it harder to understand and navigate. Queries also become more complex and the number of joins needed increases.

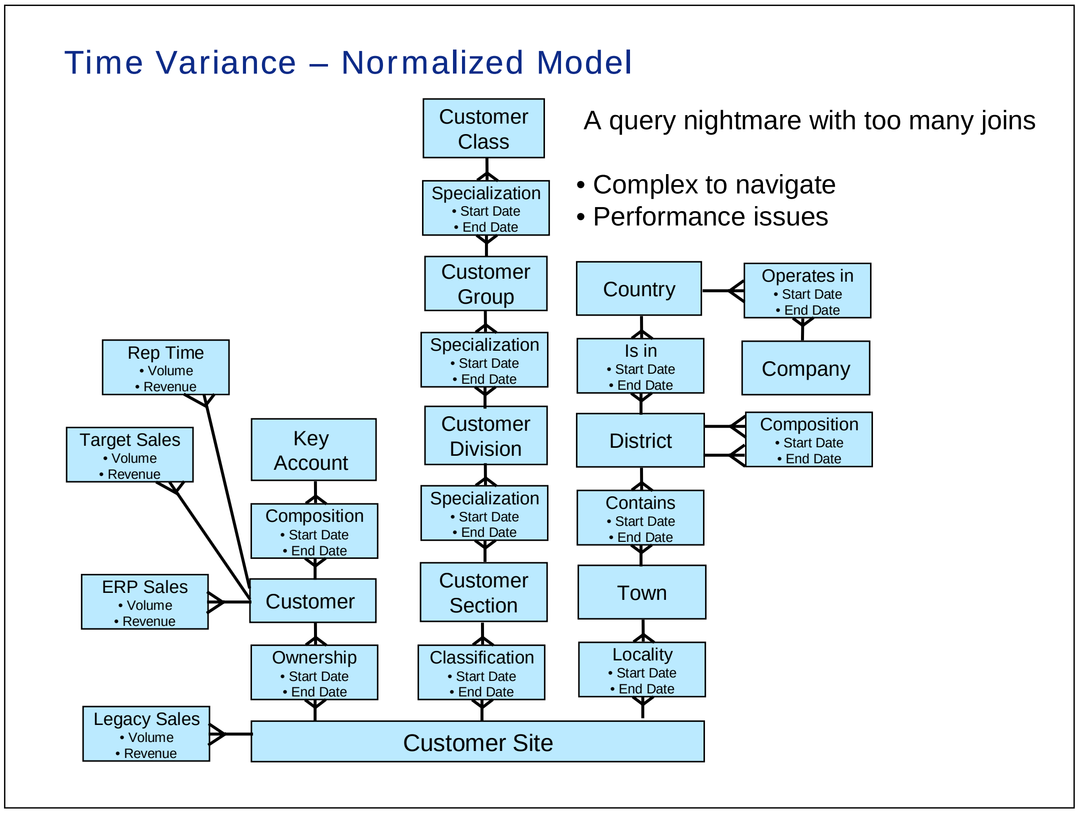
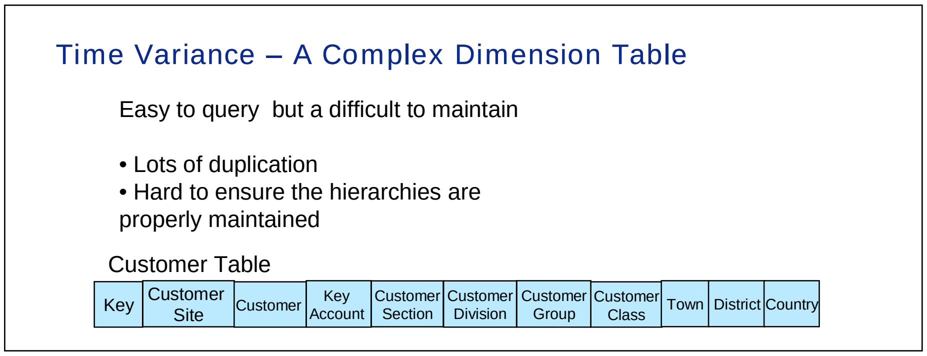


Figure 19: Handling time variance in a normalized design

The star schema model (Fig. 20) also becomes complex but for different reasons. The dimension table is easy to report from but is more difficult to manage. After deciding which attributes or relationships must be time variant, you implement a Type-1, 2 or 3 dimension:

- Type 1 is no time variance and is the most common solution.
- Type 2 is the creation of new records every time any data changes. A new surrogate key is created for each of these records and the transaction data references these keys. This is the time-variant option.
- Type 3 is where some fields are chosen to have history and these are stored in the records as additional fields (optionally with dates).



Time Variance – A Complex Dimension Table

Easy to query but a difficult to maintain

• Lots of duplication
• Hard to ensure the hierarchies are properly maintained

Customer Table

Key	Customer Site	Customer	Key Account	Customer Section	Customer Division	Customer Group	Customer Class	Town	District	Country

Figure 20: Handling time variance in a star-schema design

There is a major drawback to this technique. The choice you make at design time is the only way you can view the data. If you choose Type 2 then you have to view the data as it was at the time of the data collection, if you choose type 1 then it is always restated according to the current hierarchies. What is needed is to be able to choose. This is sometimes referred to as Type 6 (which combines 1, 2 and 3) but is even more complex to handle.

In KALIDO, time variance is automatically built in for all associations and attributes. Like with Type-6 dimensions, KALIDO enables you to choose how you would like to view the data at the time of the query rather than at design time.

5.6 Unbalanced Hierarchies

Unfortunately the world is not as well organized as computer system designers would like. In particular, hierarchies typically have varying levels. It is common to force data into fixed hierarchies so that IT systems can deal with them easily. However this means that redundant definitions have to be made and maintained.

The business requirement is shown to the left (Fig. 21) with some companies being classified as Transport companies but others as more specialized classifications.

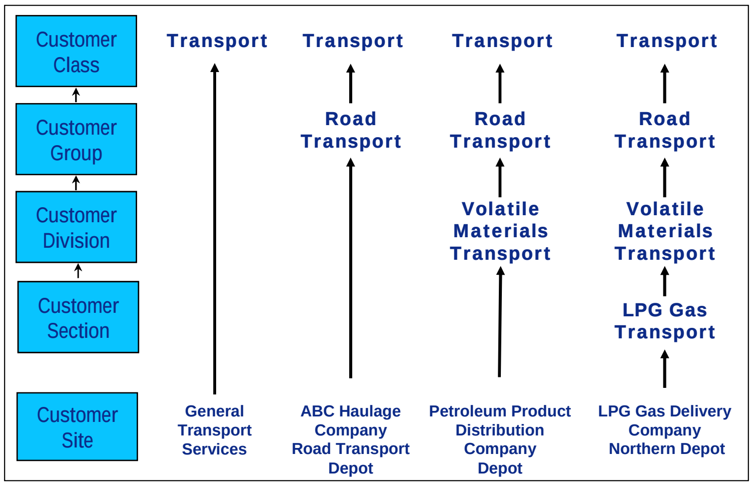

Customer Class
Customer Group
Customer Division
Customer Section
Customer Site
Transport
General Transport Services
Transport
Road Transport
ABC Haulage Company Road Transport Depot
Transport
Road Transport
Volatile Materials Transport
Petroleum Product Distribution Company Depot
Transport
Road Transport
Volatile Materials Transport
LPG Gas Transport
LPG Gas Delivery Company Northern Depot

Figure 21: Unbalanced hierarchy requirement example

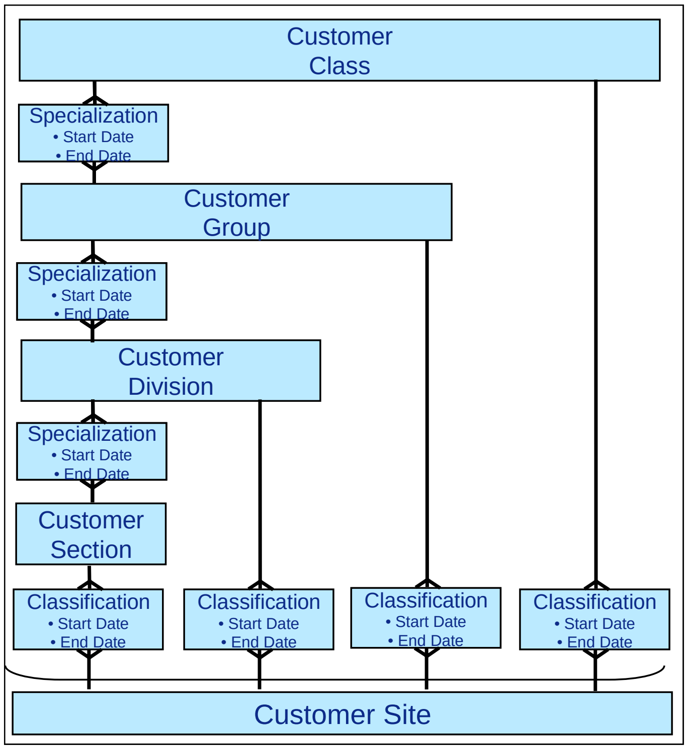
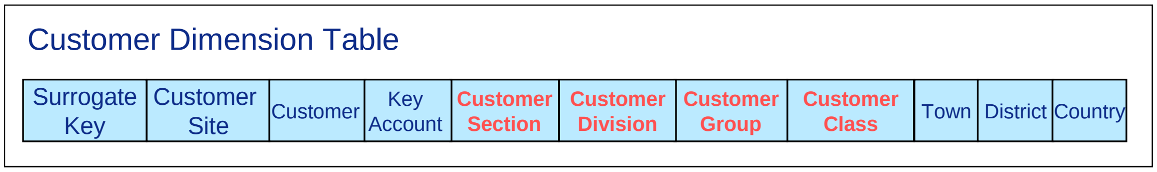


Figure 22: Handling an unbalanced hierarchy in a normalized design

This further complicates the normalized design as shown in Figure 22—especially if time variance is required, as the diagram shows. The logical model shows conditional optional relationships. This is not easy to navigate for most reporting tools as you need to know the number of levels in the branch of the hierarchy. No data duplication is required.

Using a star schema with unbalanced hierarchies (Fig. 23) is easy as long as you know the maximum number of levels. You leave the unused columns Null but you still have to manage the duplication of data. In this Customer dimension example, you would have to maintain the hierarchy in every Customer Site record if a change occurs at a higher level in the hierarchy. If you want to support time variance, this becomes a very complex structure to maintain with combinations of surrogate keys and multiple table entries.



Customer Dimension Table

Surrogate Key	Customer Site	Customer	Key Account	**Customer Section**	**Customer Division**	**Customer Group**	**Customer Class**	Town	District	Country

Figure 23: Handling an unbalanced hierarchy in a star-schema design

KALIDO automatically takes hierarchy depth into account, and manages these structures in a time variant manner.

5.7 Alternate Identifiers

Information from disparate systems commonly uses different IDs to refer to the same business object. Traditionally, ETL processing converts the IDs to a standard code or a surrogate key.

In a normalized design (Fig. 24, below) it is quite possible to have multiple unique indexed fields to identify an object. Care must be taken to reference the correct ID when loading data or querying against these objects.

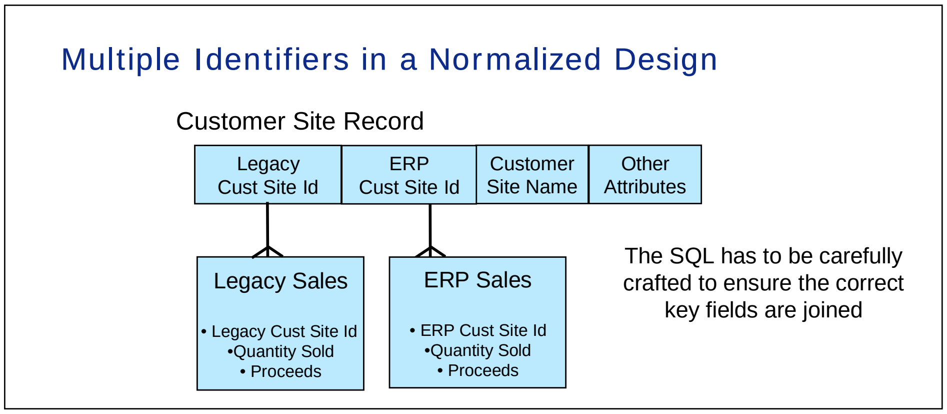
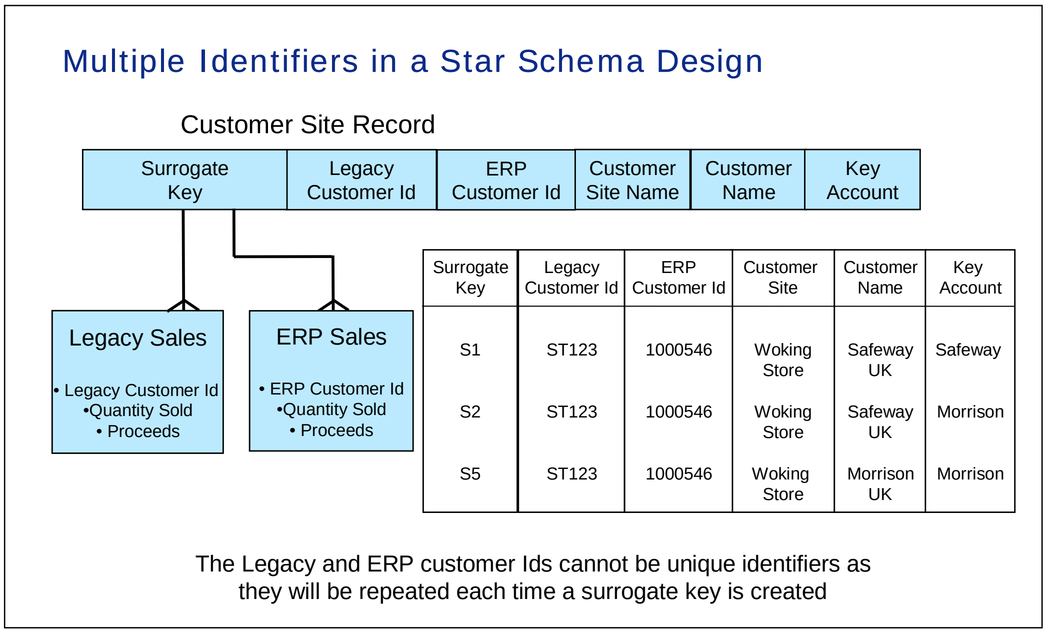


Figure 24: Handling multiple identifiers in a normalized design

In a star-schema design (Fig. 25), particularly with slowly changing dimensions, this is more difficult because the fact tables must reference the surrogate key, which may be different for the same object depending on the amount of change in its attributes. The original system IDs can be maintained in the dimension tables but in general these are lost in the ETL processes and so cannot be used in queries.



Surrogate Key	Legacy Customer Id	ERP Customer Id	Customer Site	Customer Name	Key Account
S1	ST123	1000546	Woking Store	Safeway UK	Safeway
S2	ST123	1000546	Woking Store	Safeway UK	Morrison
S5	ST123	1000546	Woking Store	Morrison UK	Morrison

Figure 25: Handling multiple identifiers in a star-schema design

In KALIDO, surrogate keys are used but these do not change with attribute or hierarchy changes and the mapping is carried out in a common way whichever code is used. All the codes are stored in the database so that they can easily be used in reporting. It is one other thing you don't have to worry about when using a KALIDO data warehouse

5.8 Business Segmentation

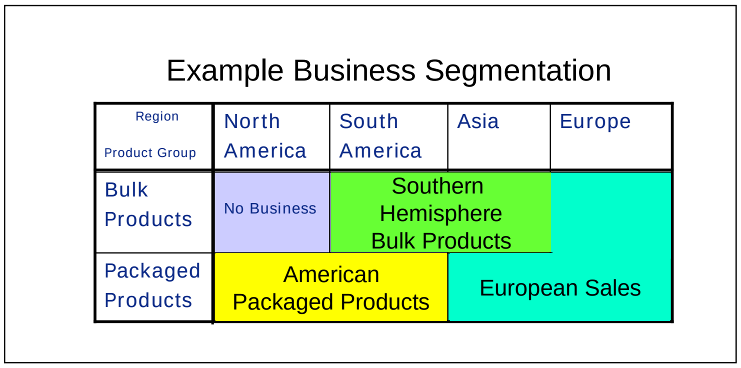
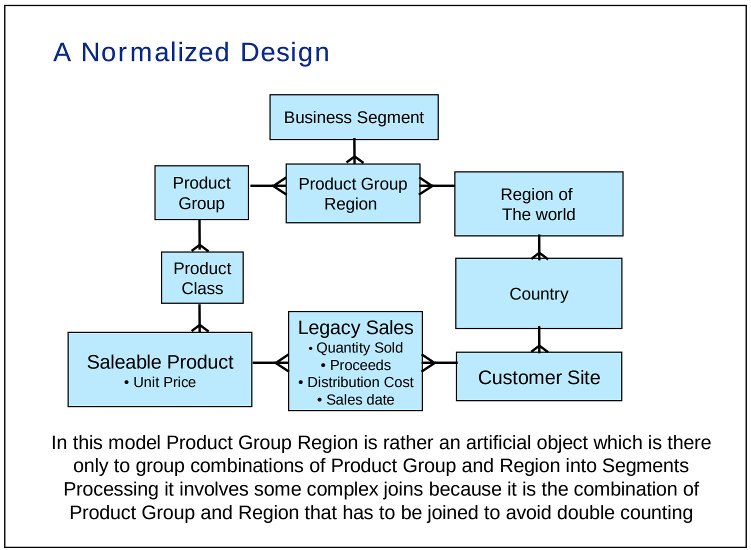


Figure 26: How a business may be segmented

Many multinational companies segment their businesses based on matrices. Segments may, for example, be based on Customer, Product and Channel or indeed in more dimensions that this. It is sometimes difficult to describe these multiple dimensions so Figure 26 shows segments defined in terms of product groups and regions of the world.

Setting up queries to operate on such a matrix is tricky, and if targets are set according to these segments then the models become complex too.

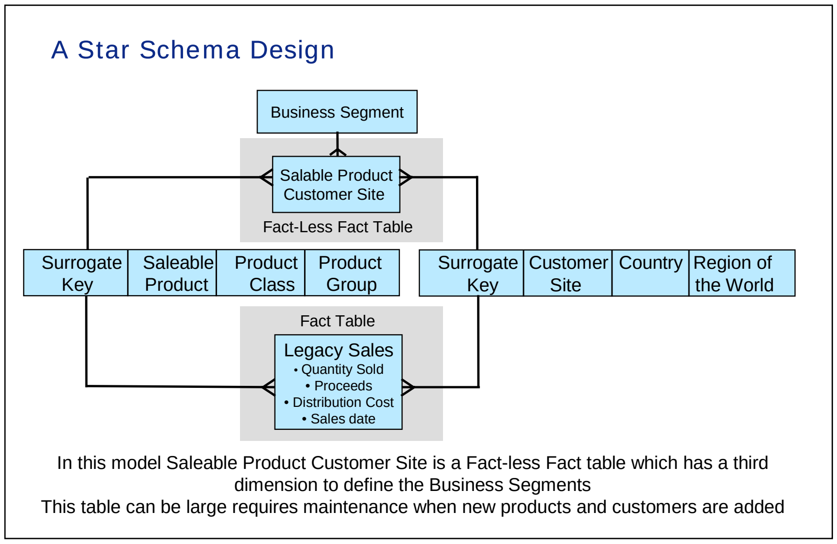


In this model Product Group Region is rather an artificial object which is there only to group combinations of Product Group and Region into Segments Processing it involves some complex joins because it is the combination of Product Group and Region that has to be joined to avoid double counting

Figure 27: A normalized model of the segmented business

In a normalized design a model such as the one in Figure 27 could be used. This model does not include time variance or multiple granularities and so is a simplification of a real business need.

A common way to resolve this in a star schema (Fig. 28) is using fact-less fact tables. This is complex to set up although it is easier to query than the normalized design. It is not essential to have it at the lowest level of granularity but the query gets more complex with multiple levels and time variance.



In this model Saleable Product Customer Site is a Fact-less Fact table which has a third dimension to define the Business Segments
This table can be large requires maintenance when new products and customers are added

Figure 28: A star-schema representation of the segmented business

In KALIDO, business segmentation is defined as a mapped Business Entity with a set of rules to determine which segment a particular transaction applies to. It uses a virtual mapping table to achieve this which is driven from the rules and is similar to the fact-less fact table. It is managed by KALIDO at multiple levels and with full time variance.

6 Benefits of KALIDO Business Modeling

KALIDO implements model changes at the conceptual level and uses a highly stable logical and physical model for storing time-variant business model metadata and reference data together. In short, the business model provides a great deal of semantic information with which KALIDO drives the whole data warehouse. By analyzing the business model as data and knowing that the data within the data warehouse is always consistent with the model, Kalido's patented approach is able to provide dramatic enterprise data warehousing benefits:

- Communication with business people is eased as discussions are about the objects that they understand rather than technical constructs of the IT world.
- Production data warehouses are created iteratively— typically within 8-12 weeks (versus the 16 months typically required to build an enterprise data warehouse).
- KALIDO data warehouses rapidly respond to changing business requirements and conditions.
- The annual manpower ownership costs of KALIDO data warehouses are often less than half that of traditional enterprise data warehouses due to the speed and ease with which functionality is changed and extended.
- KALIDO data warehouses automatically preserve historic context for trend analysis and audit reporting.
- KALIDO aids regulatory compliance by ensuring data traceability (because it only allows information that conforms to the business model to be included in the data warehouse) and by maintaining data in its true historical context with detailed logs of all processes ensues accurate data lineage and true referential integrity.
- KALIDO easily implements even very complex types of business segmentation to support management and measurement of highly matrixed organizations.
- KALIDO business models enable large companies to define corporate standards for product categories, profit calculation and market segments, etc. while enabling local business unit autonomy and variation to co-exist.
- Information can be viewed from different perspectives (e.g., by country, by brand, by customer, etc.) based upon the same underlying data—a single version of the truth presented in many different contexts. Gone are the drawn out discussions about whose figures are correct and the days where the sum of each division's profit adds up to more than the company total.

For more information on KALIDO

If you would like to learn more about business modeling or KALIDO, please visit www.kalido.com to read other white papers and to see KALIDO product demos. You can also contact Kalido and request a meeting to discuss the principles of business model-driven data warehousing and find out whether you can benefit from it. We also encourage you to evaluate Kalido's enterprise-scale Master Data Management application (KALIDO MDM), which is developed based upon the same generic modeling and adaptive data store principles.

For more information please contact us:

www.kalido.com
info@kalido.com

www.kalido.com

Kalido
1 Wayside Road
Burlington, MA 01803
USA
Tel: +1 781 202 3200

Kalido
1 The Strand
London
WC2N 5AB
United Kingdom
Tel: +44 (0) 20 7484 2200

KALIDO